ICT For Internet And Multimedia
Department of Information Engineering

Recommender System Based on Process Mining

Master Thesis

Fatemeh Nikraftar Khamene

2014329

STUDII PADUANI

XXII

Abstract

Automation of repetitive tasks can be achieved with Robotic Process Automation (RPA) using scripts that encode fine-grained interactions with software applications on desktops and the web. Automating these processes can be achieved through several applications. It is possible for users to record desktop activity, including metadata, with these tools. The very fine-grained steps in the processes contain details about very small steps that the user takes. Several steps are involved in this process, including clicking on buttons, typing text, selecting the text, and changing the focus. Automating these processes requires connectors connecting them to the appropriate applications. Currently, users choose these connectors manually rather than automatically being linked to processes.

In this thesis, we propose a method for recommending the top-k suitable connectors based on event logs for each process. This method indicates that we can use process discovery, create the process models of the train processes with identified connectors, and calculate the conformance checking between the process models and test event logs (unknown connectors). Then we select top-k maximum values of the conformance checking results and observe that we have the suitable connector with 80% accuracy among the top-3 recommended connectors. This solution can be configurable by changing the parameters and the methods of process discovery and conformance checking.

Acknowledgements

It is with deep gratitude and sweet appreciation that I, Fatemeh Nikraftar Khamene, the author of this thesis, wish and express my utmost gratitude and deep appreciation to all the people who have contributed greatly to the development and shaping of this thesis.

Andrea Burattin, Associate Professor, Software and Process Engineering at DTU
my supervisor professor, who has helped me tremendously with my thesis, always giving me advice and directing me to achieve better results.

Pietro Zanuttigh, Associate Professor, Information Engineering at UNIPD
my supervisor professor, who provided indispensable assistance in guiding me towards graduation, always responded to my inquiries and adeptly directed me towards optimal pathways.

Michal Sroka, Senior Applied Scientist, Microsoft
my co-supervisor, who helped me provide the data I needed and always provided his expert insights and feedback during my development of the thesis.

Mohammadreza Fani Sani, Senior Applied Scientist, Microsoft
for helping me with his valuable suggestions during the solution and evaluation phases.

Gemma Di Federico, Ph.D. student, Software and Process Engineering at DTU
for helping me with different parts of the thesis and also supporting me as a friend.

Contents

1 Introduction

A process mining approach is an emerging field in data science that employs an organization's transactional digital footprint to examine its business processes and identify process challenges. A process mining study looks at business processes as they are exposed to digital data stored in various systems through the metaphor of an X-ray. Process mining provides several techniques to extract actionable knowledge and insights into a process based on historical execution data[1].

Global business environments are highly competitive, requiring enterprises to turn to technology to maintain leading market positions. To perform process mining, all that is required is transactional data, along with the date and time stamps. This data is available for most companies that have an IT system supporting their business. Understanding how processes are executed is critical to maximizing automation's value. Robotic Process Automation (RPA) is rapidly gaining traction as an essential method for automating office processes. Using RPA tools to automate business processes makes it easy to have software "robots" that can trigger responses, manipulate data, and communicate[2].

A comprehensive view of all user activities in a process is necessary in order to automate office processes efficiently. Understanding how processes are executed is critical to maximizing automation's value. Using automating tools, users get a deeper understanding of the way people work and can discover which time-consuming processes can be automated. Using one of these process mining tools, Microsoft Process Advisor [3], you can easily identify common workflows and convert them into flows. By recording the tasks that make up a business process, the desktop tool creates a process diagram. A recommended set of automation can then be delivered using this information. A new way of capturing and documenting how tasks are carried out can then be created and shared with others within the organization.

Users can record desktop activity, including metadata, with Microsoft Process Advisor. There are details for very small steps that the user takes in the very fine-grained steps. These steps include clicking on buttons, typing text, selecting the text, selecting menu option, sending keys , etc. These steps together create the processes for an activity which can be done automatically. Several applications can be used for automating these processes, like using email applications for automating sending the emails or using calendar applications for automating scheduling the meetings, etc.

One obvious thing is that the processes should be linked to these applications. These applications that should be connected to the processes are called connectors. For each process, several connectors can be used, and users, based on the tasks done in the processes, select these connectors manually. There could be several ways to automate selecting the appropriate connectors for the processes, such as machine learning and data mining.

In this thesis, we present a framework for recommending the top-k best-related connector for the event logs of a process. We used recorded data published by Microsoft and labelled the cases in the event logs with known connectors. The methods that were employed are process discovery and conformance checking.

Process mining techniques can be categorized into three categories: process discovery, conformance checking, and process enhancement. In the context of process discovery, businesses can study the underlying structure of existing processes using data logs, employee insights, and other documents that provide detailed insights into the process. Process discovery is intended to make processes explicit so that everyone in the process can understand them and share them.

Using conformance checking techniques, you can observe how well an event log and a system net fit together. It is possible to check conformance for a variety of reasons. The first reason for using this method is to audit processes to determine whether reality conforms to some normative or descriptive model. The second advantage of conformance checking is that it can be used to evaluate the results of process discovery [4].

Considering a large number of options for an item, it is difficult to determine what might be a good choice without thoroughly checking out the available options. Using recommender systems, this task can be automated and performed in an efficient manner. By using process mining techniques and having train and test event logs, we can have trained process models and use conformance checking to allow us to assess to what degree such process models and test event logs process correspond to one another. So we can have a recommender system to recommend the best identified processed models for an event log.

By having models of the tasks processes and using different methods for different steps, this solution should not only be configurable but also allow users to use it to recommend what previously known tasks were performed during the new process.

1.1 Goals

A key aim of this thesis is to validate a solution that recommends the best identified processed model in order to find out which models are appropriate for the definition of an event log based on the information provided in the event log. The proposed solution should be configurable and allow users to use it for similar, varying other problems and also use different methods for different steps of this method.

1.2 Thesis Outline

Chapter 2: Problem Domain
This chapter presents and describes the problem that the thesis aims to solve.

Chapter 3: Related Works
A summary of the existing work for the different components of the thesis solution will be presented in this chapter.

Chapter 4: Problem Statement & Proposal
In this chapter, the main problem of this thesis is defined, and the research questions proposed to solve the problem are provided.

Chapter 5: Methodology
The methodology used during the development and evaluation phases of the project is discussed in this chapter.

Chapter 6: Solution
This chapter provides a detailed description of a solution to conceptualize our thesis' scope.

Chapter 7: Evaluation
Throughout this chapter, the implemented solution is tested and evaluated, providing insights into its process, reflections, and limitations.

Chapter 8: Conclusion and Future work
In this chapter, the thesis's achieved goals are discussed as well as possible improvements for future work.

2 Problem Domain

To improve performance, reduce costs, and minimize errors, many organizations are automating their manual processes in the context of digital transformation. It has been noted that process automation and process mining are key technologies for digital transformation [5]. The use of process mining can be an effective complement to process automation and provides the required transparency for digital transformation [6].

An emerging process automation approach, robotic process automation (RPA), uses software robots to automate human tasks. A virtual bot automates the execution of a process workflow after recording the actions performed by humans in the graphical user interface [7]. The system allows organizations to automate repetitive clerical tasks using scripts that encode interactions with desktop and Web-based applications. Clerical tasks include opening files, selecting fields in Web forms or cells in spreadsheets, copy-pasting data between fields and etc. [8]. Using RPA can be challenging, and you can face some key challenges such as choosing the wrong process, lack of suitable infrastructure, working with a third party, etc. By analyzing the historical behavior of a process, you can have a jump start on what and how to automate.

There are several ways to automate processes and too many tools within the RPA area. One of these tools is Microsoft Process Advisor, a service that allows you to create automated workflows. A comprehensive view of all user activities in a process is necessary in order to automate office processes efficiently. Understanding how processes are executed is critical to maximizing automation's value. Using such tools, users get a deeper understanding of the way people work and can discover which time-consuming processes can be automated. Users can record desktop activity, including metadata, with Microsoft Process Advisor. There are details for very small steps that the user takes in the very fine-grained steps. These steps include clicking on buttons, typing text, selecting the text, and changing the focus.

To automate the processes, these tasks should be connected to the applications through connectors. A connector[1] is a proxy or a wrapper around an API that allows the underlying service to communicate with Microsoft Power Automate, Microsoft Power Apps, and Azure Logic Apps. Users can create apps and workflows by connecting their accounts and using prebuilt actions and triggers [9]. There are over 275 prebuilt connectors and thousands of prebuilt templates available. Connectors, actions, and templates allow for easy integration of popular apps and services in workflow development [10].

One of the challenges of automation is selecting the suitable connector among the connector lists after recording the process. Without having knowledge of the process, selecting the connectors is complicated. The problem, in this case, is that you should manually select a connector for each step of your process, and you do not get any recommendations for choosing the best connector for your task. We need a recommender system to suggest the top-k best connectors for automating the processes. For making this recommender system, we need to discover the processes and features to find the similarities between the test and train logs.

With this recommender system, users do not waste time searching for the best connector; they just select the best connector among the recommendations and go through the au-

[1] https://docs.microsoft.com/en-us/connectors/connector-reference/

tomation of the process. Additionally, when someone does not know anything about the process and did not do the task himself/herself and wants to automate it if the names of the activities are not clear and obvious, it is challenging to select the appropriate connector, so this recommender is needed.

3 Related Works

In this project, we present a recommender system to find the suitable model among discovered models for our event logs based on process mining techniques. For this reason, in this section, firstly, we briefly discuss recommender systems and process mining techniques. Secondly, we report some related works within the recommender system in the process mining area.

3.1 Recommender Systems

Increasingly, data can be collected about user buying behaviors due to the widespread use of web-based transactions. Users' profiles, interests, browsing, buying, and rating information are included in this data. To recommend products or services based on these data, it makes sense to use this data in order to make recommendations to clients.

Each user-item pair in the recommendation problem has a utility value. In this case, the utility values of n users and d items are reflected in a $n \times d$ matrix D. It is also known as the utility matrix. Users-item pairs can be categorized based on both their buying behavior and their ratings. Recommendations should be based on these specified values [11].

These recommendation engines are designed based on the domain and the specific characteristics of the data available. Analytic technology is used to compute a user's probability of purchasing one of the products at each location so that they can receive recommendations based on these data. Generally, recommendation systems are classified as collaborative filtering (CF) or content-based filtering (CB) [12].

A collaborative filtering system analyzes historical interactions, whereas a content-based filtering system analyzes profile attributes; a hybrid technique combines both (Fig 3.1) [13]. Real-world examples of industry-strength recommender systems include the recommendations for books on Amazon and movies on Netflix.

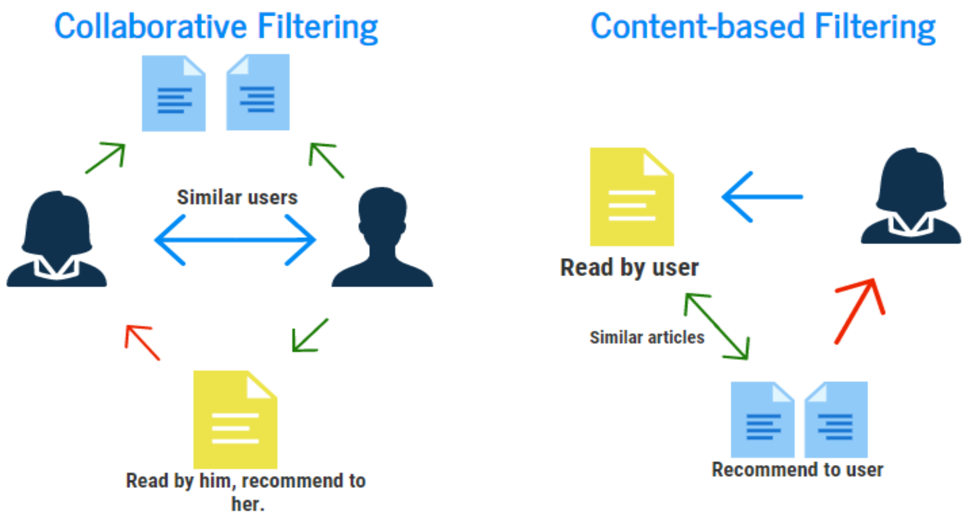


Figure 3.1: Illustration of the recommendation system types

3.2 Process Mining

A process mining analysis provides both data-oriented and business-process-oriented analysis at the same time. In process mining, three major subfields exist: process discovery, conformance checking, and operational support [1].

A process discovery method is intended to help identify the underlying process captured in the event data and create a model of that process. Described process behavior according to the specifications of a particular process modelling language forms a process model. Process discovery algorithms take an event log as input, and they then generate a descriptive process model as an output which contains control-flow information across activities in different notations.

In process modelling, there are two types: declarative (defining behavior that is not permitted) and procedural (defining behavior that is permitted). Modelling languages such as Unified Modelling Language (UML) activity diagrams, event-driven process chains (EPC), Business Process Model and Notation (BPMN) and Petri nets [14] are among the second type. Declare, Dynamic Condition Response (DCR) Graphs and Case Management Model and Notation (CMMN) [15] are examples of declarative modelling languages.

A second task is to perform a conformance check, which analyzes the degree of compatibility between event data and a process model. A better reflection of reality can be obtained by exploiting the differences between the discovered/reference process model and the corresponding event log. The following four quality dimensions are typically measured when assessing conformance [16]:

- Fitness: the found model should allow for the behavior seen in the event log;
- Precision: according to the discovered model, behavior should not be totally different from what is seen in the event log;
- Generalization: the discovered model should generalize beyond the example behaviour shown in the event log;
- Simplicity: the best model is one that is as simple as possible.

Taking a look at the first dimension modelling language in Dunzer et al. [17], Petri nets are by far the most frequently used process modelling language in conformance checking literature. The Petri net is a visual representation of execution logic such as restrictions, control flow, and concurrency. Petri nets have been developed as a formal process modelling language because of their mathematical background, their ability to capture concurrent behavior, and their function as state charts. As an example of this, Burattin et al. use Petri nets as a representative language for all procedural modelling languages in their framework for conformance checking [18].

According to Carmona et al.[19], and van der Aalst [20], log replay algorithms and trace alignment algorithms are two general approaches to algorithms addressing these problems. As part of the log replay algorithm, the model and the log are interpreted, and then each trace is rerun, event by event, on the model. A conformance metric can then be determined using different computing techniques. Token-based log replay in Rozinat et al. [21] is an example of such a technique. Unlike log replay algorithms, trace alignments are capable of expressing deviations and conformance on an event level.

Lastly, operational support aims to improve or enhance process mining results, for example, by directly mapping bottleneck information onto a (given) process model. This subfield of process mining involves enriching the process model by adding emphasis and supplying further information (e.g., time-stamps, KPIs, suggestions) to enhance the process model [22].

3.3 Recommender Systems in Process Mining

In Wang et al. [23], the authors first summarized their previous work on using reference models to evaluate process mining algorithms empirically. As a result, the quality of a discovered model was measured by its behavioural and structural similarities with a reference model. A fraction of the process models from the given enterprise is proposed in this paper as reference models. To obtain two trained models, they applied regression analysis - one for behavioural measures and the other for structural measures. These trained models can then be applied to any process models within the enterprise, and the estimated similarities between the algorithms are obtained. In order to rank the business process mining algorithms based on their estimated similarities, their framework recommended the algorithm with the highest similarity to each of these enterprises.

A labelled Petri net was used as the modelling language, and structural similarity was measured by using dependency graphs. As the process matrix of a dependency graph, the incidence matrix of the labelled Petri net is used. The behavioural similarity was also assessed using the Principle Transition Sequence (PTS) metric. With over 90% accuracy, the best mining algorithm found by their framework is almost identical to the one found by empirical evaluation. They could use their experience to determine the best process mining algorithm based on the enterprise's process models instead of empirically evaluating each process mining algorithm.

Ribeiro et al. [24] presents a recommender system that employs portfolio-based algorithm selection strategies to find the best discovery algorithm for the data at hand. This paper offers a framework for integrated algorithm selection based on machine learning. Formally, this framework elaborates on the Algorithm Recommender System(ARS) approach. Training and recommending are two functions of this recommender system. Prediction models are built using knowledge generated by the training function.

First, the experiment results were retrieved from the repository. For each event log and measurement (performance or conformance) in the results, the ranking of discovery techniques was computed. A ranking of techniques must contain all control-flow miners used in the experiments. In the case a ranking was incomplete, a machine learning algorithm (e.g., SVM or Neural Networks) was applied to predict the missing ranking values. The features of the log were extracted for each event log in the results. For each measurement in the results, the corresponding prediction model was trained using the rankings of discovery techniques and the features of the logs. The recommending function used the prediction models to obtain the top-k best-performing discovery techniques for an event log.

Customer journey analysis is a hot topic in marketing. As a result of process mining, we can discover the process that best describes the user's behavior, obtains valuable insights, compares the processes of different clusters of users, and improves the journey by providing individualized recommendations. Terragni et al. [25] explains the new possibilities offered by using process mining on web logs to explore customer journeys, predict customer behaviors, and suggest actions to maximize particular KPIs (Key Performance Indicators). Their dataset contained click-stream information from 2 million users in a time frame of one month. For the preprocessing, they used hand written rules and unsupervised clustering proposed in Poggi et al. [26] to map each URL with a specific activity in the activity set. Using the Fuzzy Miner algorithm in Disco, they analyzed the general behavior of the web log L. Then, they analyzed the event log using manual filtering and unsupervised clustering to obtain more interesting insights.

They implemented a recommender system based on collaborative filtering techniques proposed by Hu et al. for implicit dataset [27] to improve the personalized recommendations provided to the users. The recommender system used the time spent on each product page, the visit frequency, and other journey-related factors. They divided the original data into a masked data set and an evaluation data set and checked if the users actually visited the product pages recommended to them by the system. They calculated the mean area under the curve (AUC) for each user that had at least one masked product page and compared it to the mean AUC for recommending the most popular product pages to each user.

Many process discovery techniques are available, making it difficult for practitioners to choose the most appropriate one. Analyzing event logs was studied in SKLM Broucke et al. [28], which provides valuable insights into the role of event log characteristics in process discovery. For eight different structural patterns, eight process models were designed and used to generate artificial event logs. The generated event logs had different properties. There were twelve different discovery techniques studied: Alpha miner, Alpha miner+, Alpha miner++, Heuristic miner, Genetic miner, DT(Duplicate Tasks) Miner, DWS Miner, AGNEs Miner, Causal Miner, ILP Miner and TS(Transition System) Miner. For each discovery technique, the configuration options were largely kept at the defaults. However, modifications have been made to the Heuristics Miner and Causal Miner.

The conformance checking metric was used in the third and final phase of the experimental setup to assess the quality of different mined process models. They applied the CoBeFra conformance checking benchmarking suite to facilitate the conformance checking phase. Lastly, the authors applied a non-parametric statistical approach to comparing the performance of different process discovery techniques in a robust manner as outlined in De Weerdt et al. [29] and Demšar et al. [30], including a Friedman test followed by an appropriate post hoc analysis. Under the null hypothesis of no significant differences between treatments, these non-parametric tests calculate average rankings per treatment. Therefore, this test compares the different process discovery techniques to the best-performing one. A general recommendation for choosing a process discovery technique was formulated based on general performance results, event log and model characteristics (from the regression analysis) and timing issues.

4 Problem Statement & Proposal

A process mining technique enables process stakeholders to see process information, therefore allowing them to identify the process flow using actual data. Process discovery is the most well-known process mining technique. For many organizations, it is surprising to see that existing techniques are indeed able to discover real processes merely based on example behaviors stored in event logs.

Process discovery and conformance checking can be very helpful in recommending the appropriate model to describe the event log when we have a set of discovered models with different algorithms and a set of unknown event logs. The most recent solutions that have been developed are for recommending the best algorithm among a set of algorithms which is somehow different from what we need, but we can use parts of their methods and idea.

There is also the problem where automation of the processes is needed by numerous organizations where the process must be automated. Therefore, in order to choose the best platform for processing the tasks, the platform selection for each step is made manually. Without proper insight into their processes, process stakeholders could even make wrong decisions based on false assumptions. They cannot make correct selections without knowledge of the steps involved in a process.

4.1 Research questions

To propose a solution that addresses the problem, this thesis needs to be developed based on some research questions that must be considered when approaching the thesis. The following research questions are:

RQ1: Are process mining algorithms such as Inductive Miner and Heuristic Miner able to discover desktop recorded events acceptably?

RQ2: Is it possible to use conformance checking algorithms such as alignments for recommending the best process connectors for automating the processes?

RQ3: Can we use the fitness values from conformance checking to recommend top-k best connectors?

5 Methodology

The chosen methodology for this thesis is Knowledge Discovery in Databases (KDD). In KDD, adequate, practical, and understandable patterns are identified in large and complex data sets. The KDD method is based on data mining, which involves analyzing data, building models, and discovering previously unknown patterns.

The model of the KDD process consists of the following steps in an iterative and interactive way:

1. **Understanding goal**: This is the initial preliminary step. It lays the groundwork for understanding the problem and how to proceed with the various decisions. In this thesis, the goal is to present a recommender system to recommend the most suitable model from the identified discovered models for an unknown event log.
2. **Creation of a target dataset – selection**: Once the objectives have been defined, the data to be used for knowledge discovery should be determined. Since this thesis is in collaboration with Microsoft, the data has been published by them.
3. **Data cleaning and preprocessing**: The purpose of this step is to find missing data and remove noisy, redundant, and low-quality data to enhance data reliability and effectiveness. Throughout this thesis, data was labelled, and parts of the data that were missing information were filled in manually based on the other provided information.
4. **Data reduction and projection**: Identifying useful features of the data (based on the goal), including dimension reductions and transformations, should be done in this step.
5. **Selection of data mining task**: This is the time to decide what kind of Data Mining task to use, such as classification, clustering, regression, or another task. Since we want to recommend the best model for defining the event log, we can say that our task is classification. We classified the test event logs to their most relevant model.
6. **Selection of data mining algorithm(s)**: Having chosen the technique, we can now decide which strategy to employ. A particular technique will be selected during this stage to search for patterns with multiple inducers. This thesis uses different process discovery models and conformance checking algorithms from process mining.
7. **Data mining**: Finally, the implementation of the Data Mining algorithm has been reached. Our algorithm may need to be used several times before a satisfactory result can be achieved.
8. **Interpretation of mined patterns**: After the data mining techniques have been applied and iterations have been completed, these patterns need to be represented in discrete forms, such as bar graphs, pie charts, histograms, etc., in order to examine the impact of data collected and transformed in previous steps. Furthermore, this helps determine the effectiveness of a given data model in a particular domain. The results of this thesis are shown in the tables.
9. **Consolidation of discovered knowledge**: The knowledge can now be incorporated into another system for further activity. When changes are made to the system, and their impact is measured, knowledge is made effective. By completing this

step, KDD becomes effective. Our approach should be tested on Process Advisor, but since it takes time to do so, it is not included in this dissertation.

Fig 5.1 represent the spiral model of some part of our methodology. The first iteration took about two months since the problem was not indicated properly, and the data was not published completely. Each second and third iteration took about one month. The spiral model has four main steps. A detailed explanation of each step is indicated below.

- **Plan**: The data processing (third and fourth steps of methodology) was carried out during this phase to plan the most effective solution for our problem. In the process of processing data, two main steps were performed: Labeling the steps in event logs and splitting the data for training and testing. A total of two iterations were needed to complete the steps.
- **Design**: As part of this phase, we completed the sixth step of the methodology. In this step, we selected algorithms for process mining techniques such as process discovery and conformance checking after studying the data.
- **Development**: During this phase, the implementation of the algorithms (step seven of the methodology) was accomplished by coding. To implement our code, we used PM4PY, a python library designed for process mining approaches.
- **Evaluation**: This phase consists of evaluating the results and interpreting patterns (step eight of the methodology). Looking at each iteration's results allowed us to plan the next iteration to improve the results. The conformance checking fitness values were the metrics for evaluating the results.

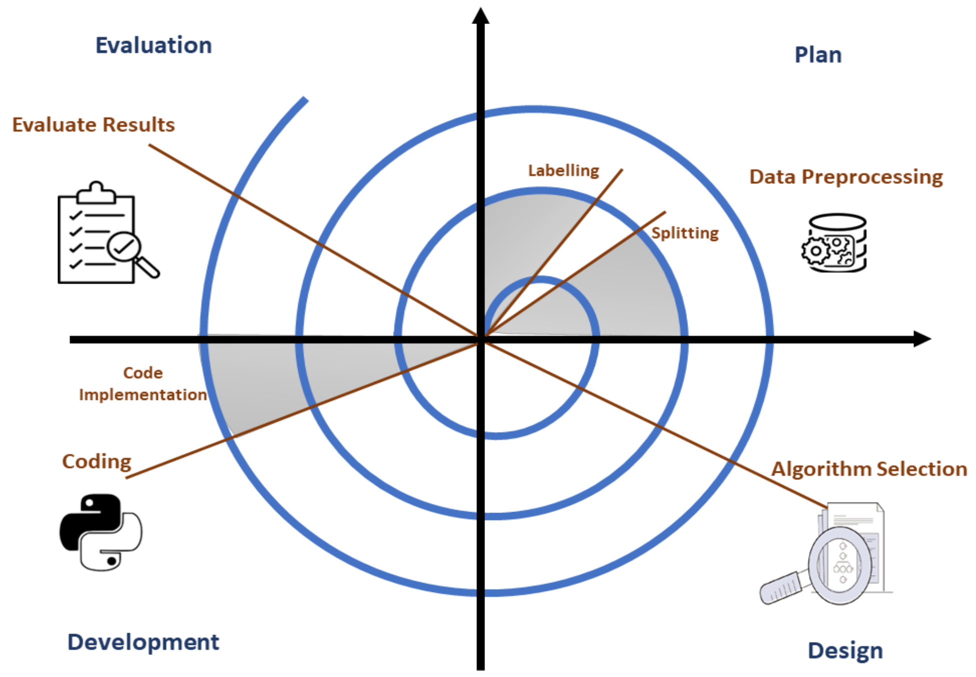


Figure 5.1: Spiral model of the project

6 Solution

In the following sections of chapter 6, the creation of the recommender system framework will be discussed. The first section shows an overview of the project domain and solution and an introduction to the following sections. An introduction to the various important algorithms of our solution is presented in the second section of this chapter. The third section of the chapter will go over the solution implementation of the system. At the end of this chapter, we provide the implementation of our solutions by showing them as algorithms.

6.1 Solution overview

As mentioned in chapter 1, the data is provided by Microsoft Process Advisor. The data we use are event logs containing various information from monitoring and recording a user's desktop while performing tasks.

These tasks are becoming increasingly automated these days, so some platforms and applications are needed to automate them. The event logs of the recording processes need to be connected to the connectors (platforms and applications). Hence, the first thing we have to do is to understand which event logs are related to which connectors in order to automate them.

There are many ways to use such as supervised and unsupervised machine learning algorithms to classify the logs. However, the use of machine learning is unnecessary when simpler and equally effective alternatives are available. Machine learning models are a black box. Despite knowing the underlying mechanics behind these models, it's still not very clear how they get their final results. Based on these assumptions, we found a more straightforward way to classify the event logs and recommend the best connectors for them based on the ranking results. Here you can see the overview of the recommender system in Figure 6.1.

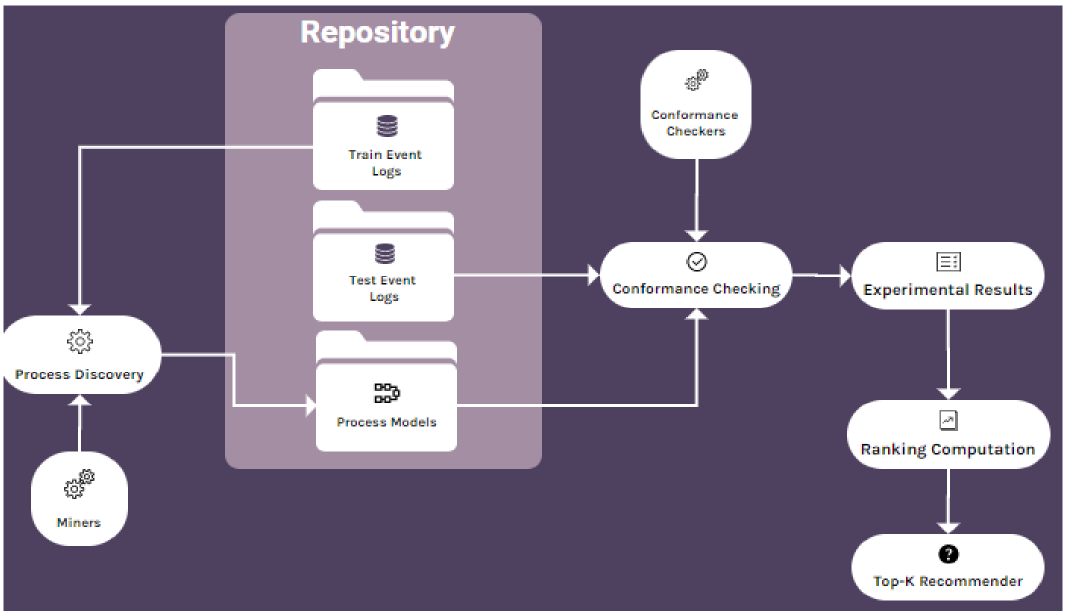


Figure 6.1: An overview of the recommender system

The recommender system takes the train and test logs as input and returns the connector name, the fitness value of conformance checking and the ranking of that result among the fitness results.

As we can see, everything starts with the data. In order to determine the appropriate methods and algorithms to apply, we must first understand the structure of the data. For our solution to be developed, we need the event logs recorded from desktop activities. A detailed description of the data and the required preprocessing is provided in section 7.1.

To test our solution and see its results, we should split these event logs into two groups: train and test. Using process discovery, we make a process model for each connector using the train data. Following that, the test data are used to calculate the conformance checking between them and the process models. Based on the results of the conformance checking on all the test event logs, we can identify the top-K maximum values of the conformance checking for each log and match each log with the process models according to their ranking.

Section 6.2 contains a complete description of the algorithms, including process discovery and conformance checking. A comprehensive review of the implementation is given in section 6.3, including how the recommended rankings are determined.

6.2 Techniques

Our approach can be used to build a recommender system, with event logs as input objects and discovery techniques as algorithms. The process-mining spectrum is quite broad, and this thesis focuses on process discovery and conformance checking.

6.2.1 Process Discovery

The process discovery process uses event data to learn process models. By using an event log, a discovery technique can produce a process model without requiring any additional information. The α-algorithm, for instance, produces a Petri net to explain the behavior recorded in the event log [31].

During the discovery experiment, a control-flow algorithm is applied to an event log in order to produce a model of the process [24]. Having discovered a process model from an event log, it can be used for a variety of purposes, such as performance analysis, compliance checking, and predictive analytics [31]. Numerous techniques have been proposed for process discovery over the last two decades.

Increasingly, new process discovery methods have been proposed since 2010. The Leemans et al. [32] family of inductive mining techniques can be used as an example. These discovery techniques (IM, IMf, IMc, IMd, IMfD, IMcD, etc.) produce well-structured models and are highly scalable. The result of published results on how discovery algorithms were evaluated [29, 33, 34, 28, 35], led us to choose Heuristic Miner [36] and Inductive Miner [32] algorithms. This selection is also due to the evaluation method where we use PM4PY, which is limited to some algorithms.

One of the most significant techniques in process mining is the Alpha Miner algorithm. Alpha Miner algorithm discovers a workflow model from the local dependence between events based on Petri nets. The Alpha algorithm assumes that event logs contain no noise and are complete with respect to all allowable binary sequences. This means that the α-algorithm is sensitive to noise and the incompleteness of event logs.

An extension of the formal α-algorithm is Heuristic Miner. There are two main steps in the HM algorithm. The first step is to create a dependency graph. When building the

Petri net model, this dependency graph will contain all the causal dependencies. When calculating the strength of causal relations, the Heuristics Miner takes into account the frequencies of the basic ordering relations. The algorithm creates one dependency on the most suitable causal successor and predecessor of a given task by default. The next step is to determine the semantics of the split/join points in the dependency graph. Based on the frequencies of the dependencies, the type of split/join is determined. A Heuristics Miner can handle noise and express the main behavior (i.e., not all details and exceptions). Except for duplicate tasks, it supports mining of all common constructs of process models (i.e., sequence, choice, parallelism, loops, invisible tasks) [35, 36].

The process of IM involves, recursively, selecting the root operator that best describes log L, dividing the activities in log L into disjoint sets, and then using these sets to split log L into sublogs. As a result, sublogs are mined recursively until they contain just one activity in a sublog [37]. By adding infrequent behavior filters to all steps of IM, Inductive Miner - Infrequent (IMf) was introduced. When IM is used, frequencies of traces and events are ignored, whereas IMf measures frequencies to distinguish between frequent and infrequent behaviours.

Some of the advantages and disadvantages of these mining algorithms are shown in table 6.1.

Table 6.1: Advantages and disadvantages of the mining algorithms

Alpha	Heuristic	Inductive
Cannot handle loops of length one and length two	**Takes frequency into account**	**Can handle invisible tasks**
Invisible and duplicated tasks cannot be discovered	**Detects short loops**	**Model is sound**
Discovered model might not be sound	**Does not guarantee a sound model**	**Most used process mining algorithm**
Weak against noise		

After having the process models for each connector in the train data, we go through the conformance checking part to get the result values to start the ranking part.

6.2.2 Conformance Checking

In conformance checking, differences and commonalities between an event log and a process model are identified and diagnosed [38]. It is possible to use conformance checking to determine if the recorded reality matches the model and vice versa. To measure fitness and precision, several techniques and measures have been proposed, including token-based replay [21], alignments [39, 40], etc.

Using token-based replay, a trace is matched with a Petri net model, starting from the initial point in order to discover which transitions are executed and where tokens remain or are missing. This algorithm also has some problems. Due to the handling of invisible transitions, which are included in the output models of algorithms like the heuristic miner or the inductive miner, this technique suffers from several known drawbacks. It is possible, for instance, for models to get flooded with tokens in highly non-conforming executions, which could enable unwanted parts of a process model and compromise the overall analysis of the fitness process [41].

It must be noted that a significant portion of the identified literature uses trace alignment algorithms to calculate the conformance of processes. Accordingly, trace alignments are currently considered the standard conformance checking technique [42]. Procedural process modelling languages are most commonly used in trace alignment algorithms. Unlike log replay algorithms, trace alignments can express deviations and conformance at the event level and provide a more robust conformance analysis. As a result of alignment-based conformance checking, the events of a particular case are aligned with the closest matching path permitted by the process model. Using such alignments, we are able to determine the degree of conformance between the observed events (the reality) and the process model. Using alignment-based conformance checking, fitness is evaluated by aligning the event log traces, and the visible traces of the process model [42]. A fitness metric is calculated using different kinds of cost functions in trace alignments [17].

6.3 Solution Implementation

There are several process mining tools available today, including both open-source, such as ProM and Apromore, and commercial, such as Disco, Celonis, ProcessGold, etc. Often, these process mining tools are only accessible through a graphical user interface, making them limited for large-scale experimentation. Since we have a bunch of data and it would be hard to work with software tools, we used the Python library called PM4PY.

There are many mainstream process mining techniques in the PM4Py library [43], including: Process discovery algorithms such as Alpha Miner, Heuristic Miner, Inductive Miner, Conformance Checking algorithms such as Token-based replay, alignments, footprints and assessment of fitness, precision, generalization and simplicity of process models. There are also other techniques such as graphs, filtering, and so many others which are not included in this thesis.

Based on the previous information and the PM4PY python library, we used the Heuristic Miner and Inductive Miner process discovery algorithms, and for the conformance checking, we selected alignments and token-based replay methods for our experiment. The python coding was done in Google Colab[1]. Our implementation steps are as follows:

1. Importing all the event logs (CSV files) such as train and test from the drive and converting them to the event logs with the selected caseid, activity and timestamp.
2. Creating a function with the event logs as input and configurable discovery and conformance checking algorithms. This function gets the train event log and creates a process model based on the selected model in the function, and then calculates the conformance checking between the processed model and the test event log.
3. Creating a table containing the train log names as models and the test log names, and all the mean fitness values achieved from the function.
4. Sorting and selecting top-k maximum values of the fitness values and their assigned connector model names.
5. Check if the correct connector name is among the top-k recommended connectors.

6.4 Algorithm Implementation

This section presents the implementations as algorithms for this thesis. The first subsection shows the main function for process discovery and conformance checking calculations and the second subsection represents how we used the main function for the train and test data. The last subsection provides the algorithm of the recommendation system.

6.4.1 Algorithm for process discovery and conformance checking

This function includes all the procedures of calculating the conformance checking between the model and the logs using different algorithms both for discovery and conformance

[1]https://colab.research.google.com/

checking. As indicated, there are two inputs as training and testing event logs and two other inputs as discovery and conformance checking methods. As mentioned before, the algorithms for discovery are Inductive Miner, Inductive Miner Infrequent (both with default parameters and noise threshold equal to 0.5) and Heuristic Miner (defaults parameters). Also, the conformance checking algorithms are Alignments and Token-based Replay. This algorithm first discovers the process model from the training event log, and then after calculating the conformance between the model and the test log, it returns the mean conformance checking fitness value (code A.2).

Algorithm 1: Main Function for process discovery and conformance checking

```
Input  : $\boldsymbol{TrainingEventLog}$          /* event log for training */
Input  : $\boldsymbol{TestingEventLog}$           /* event log for testing */
Input  : $\boldsymbol{Discovery}$                 /* discovery algorithm to use */
Input  : $\boldsymbol{Conformance}$               /* conformance algorithm to use */
Output: $\boldsymbol{MeanFitnessValue}$
1 $\boldsymbol{Model} \longleftarrow Discovery\ (\boldsymbol{TrainingEventLog})$
2 $\boldsymbol{Conformance} \longleftarrow Conformance\ (\boldsymbol{Model},\ \boldsymbol{TestingEventLog})$
3 return $mean(\ \boldsymbol{Conformance})$
```

6.4.2 Getting the experiment results of all train and test data

By having the train and test lists containing all the train and test logs, and discovery and conformance checking algorithms as inputs, we iterate through the train and test logs lists and apply the first algorithm 1 on the logs (codes A.1, A.3). The returned value will be a dictionary containing the conformance fitness value among the train models and the test logs.

Algorithm 2: Applying the algorithm 1 on all train and test data for getting the experiment results

```
Input  : $\boldsymbol{TrainingLogList}$           /* collection of logs for training */
Input  : $\boldsymbol{TestingLogList}$            /* collection of logs for testing */
Input  : $\boldsymbol{Discovery}$                 /* discovery algorithm to use */
Input  : $\boldsymbol{Conformance}$               /* conformance algorithm to use */
Output: $\boldsymbol{performance\ Matrix}$
1 $\boldsymbol{performance} \longleftarrow (\boldsymbol{empty\ dictionary})$
2 for $\boldsymbol{logTraining}$ in $\boldsymbol{TrainingLogList}$ do
3     for $\boldsymbol{logTesting}$ in $\boldsymbol{TestingLogList}$ do
4         $\boldsymbol{performance}[logTraining][logTesting] \longleftarrow$
            $\boldsymbol{Algorithm1}(logTraining,\ logTesting,\ Discovery,\ Conformance)$
5     end
6 end
7 return $\boldsymbol{performance}$
```

6.4.3 Recommendation algorithm

After having the experimental results, we can have the recommendation system algorithm 3 using the discovered models from algorithm 2. The third algorithm takes the list of mined models, the log for getting the recommendation, K value and conformance checking algorithm as inputs. K value indicates the number of first K elements of the list we want to have as the recommendation. In this algorithm, the conformance fitness value is calculated between the models in mined model lists and the log for the recommendation. Then we have the list of all conformance fitness values, and by sorting them, the algorithm returns the first K elements of the list of highest conformance values and the corresponding models.

Algorithm 3: Recommendation algorithm

```
Input  : $MinedModelList$  /* list of mined models starting from all training logs */
Input  : $LogForRecommendation$  /* new log for which we'd like to have a recommendation */
Input  : $K$  /* how many recommendations to get */
Input  : $Conformance$  /* conformance algorithm to use */
Output: $Model$  /* the recommended model */
1 $maxConformance \longleftarrow \{\}$
2 for $model$ in $MinedModelList$ do
3     $c \longleftarrow Conformance(model, LogForRecommendation)$
      $maxConformance \longleftarrow maxConformance \cup \{ (model, mean(c)) \}$
4 end
5 $listOfBestConformance \longleftarrow sort(maxConformance)$  /* sorts the set maxConformance according to c and generates a list with highest conformance on top */
6 return $listOfBestConformance[1..k]$ /* return the first k elements on the list, which are the elements with highest mean conformance value */
```

7 Evaluation

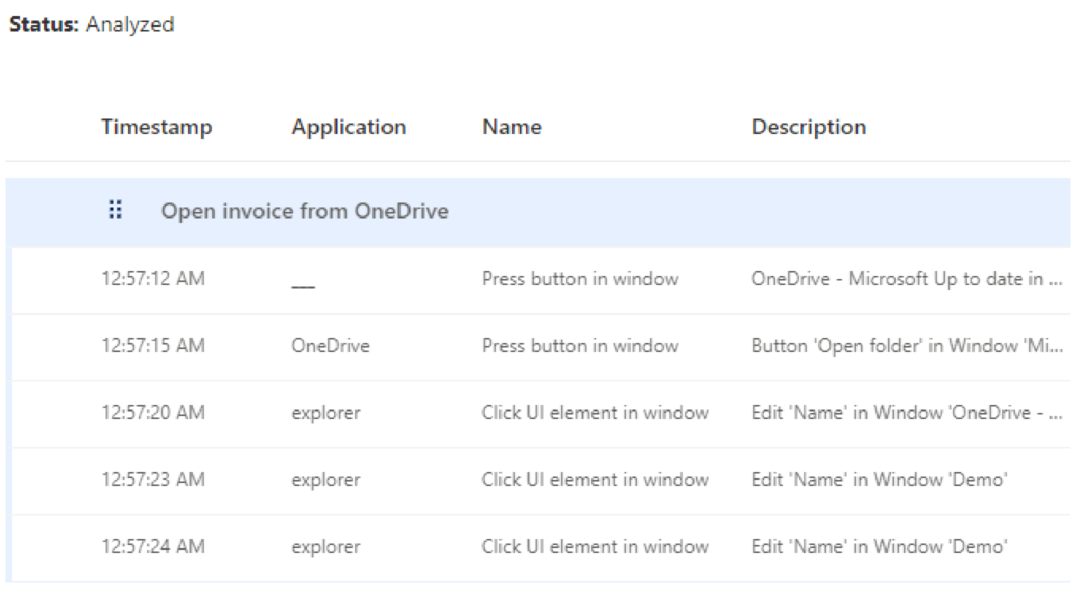
In this chapter, we show the results after our implementation discussed in section 6.3. Based on the results, we also answer the research questions mentioned in chapter 4. The first section provides a description of the data that we used for the project. The data creation process, preprocessing, and labelling of the data will be described in this section. The second section will discuss the results and outcomes.

7.1 Data Processing

7.1.1 Overview of Process Advisor

As mentioned before, the data is from Microsoft Process Advisor, which is an important component for using process mining, allowing you to identify common workflows and convert them into a flow. With this desktop tool, you can record the various steps required to complete a business process (fig. 7.1) and generate a process diagram of all those steps. A recommended set of automation can then be generated by editing the data and modifying it. This process can then be shared with other members of your organization, which provides a new way to document how tasks are completed.

Using all the relevant data from the recording, Process Advisor is able to produce a process map (fig. 7.2) that provides insight into how a process functions. This map can be used to assist you in refining your process and suggesting where it could be automated. In a process map, the activities are shown with a start and an end. A node is an activity in the map, and you can see the paths and times taken through the activity in different recordings. An automation designer tool is then available to assist with automating business processes.



Status: Analyzed

Timestamp	Application	Name	Description
Open invoice from OneDrive			
12:57:12 AM	—	Press button in window	OneDrive - Microsoft Up to date in ...
12:57:15 AM	OneDrive	Press button in window	Button 'Open folder' in Window 'Mi...
12:57:20 AM	explorer	Click UI element in window	Edit 'Name' in Window 'OneDrive - ...
12:57:23 AM	explorer	Click UI element in window	Edit 'Name' in Window 'Demo'
12:57:24 AM	explorer	Click UI element in window	Edit 'Name' in Window 'Demo'

Figure 7.1: The recorded data shown in Process Advisor

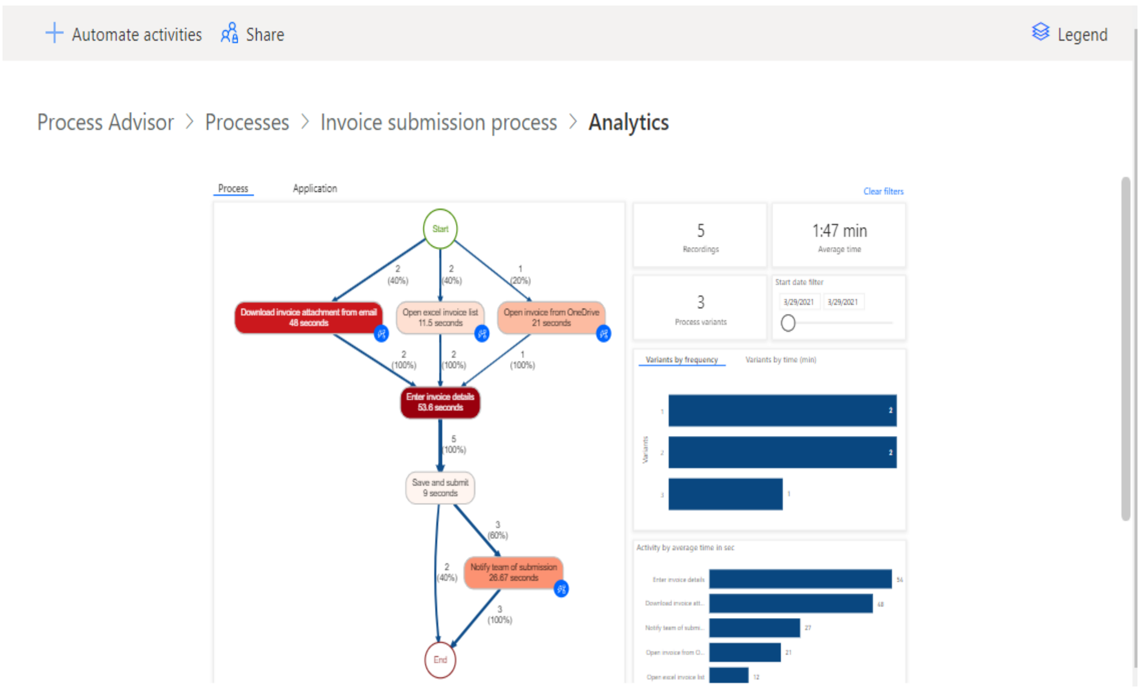


Figure 7.2: The process map shown in Process Advisor

7.1.2 Data Details

To achieve the results of our experiment, we needed to collect a lot of data in order to perform the experiments. As a result, we have a dataset published by Microsoft[1] containing daily business tasks for a period of time. The tasks were labelled based on what they were doing, such as sending an email to a colleague, checking the weather, requesting approval, etc. Data includes 50 folders containing 165 CSV event logs. Each folder contains different instances of a certain process done with different applications at different times. There are 45 folders, each with three instances of a process, and five other folders, each with six instances of a process which in total are 165 files. The published data may be different in the future since we had the first draft of data, and there can be no folders in the published data, and just the event logs can be seen.

The recorded processes are event logs consisting of different information such as recording ID, Process ID, Time Stamp, Step Name, Step Description, Application Process Name, Application Parent Window Name, Automation Step, Label Event Name and Label Event ID. The Process ID is the same for the instances of a process, and the Recording ID is unique for each event log. Also, we have 19 different Step Names where you can see the top 5 frequent ones in table 7.1.

[1] https://github.com/microsoft/50BusinessAssignmentsLog

Table 7.1: Top-5 frequent Step Names

Step Name	Frequency
Click UI element in window	2504
Press button in window	1406
Populate text field in window	718
Select menu option in window	404
Send keys	387

There is also an information file containing the related connectors for each process where the user selects the best connector for automating the process. There are more than 300 connectors now available on Power Automate, but in this experiment, we have only 25 connectors used while automating the processes (Table 7.2). There is more than one connector related to an event log, mainly two used for automation, but three as well.

Table 7.2: Connectors for the experiment

Connector	Recordings
Googlecalendar	30
Approvals	30
Onenote	24
Office365users	21
Microsoftforms	21
Outlook	21
planner	21
Sharepoint	18
Sendmail	18
Rss	18
Teams	18
Excelonlinebusiness	15
Conversionservice	15
Powerbi	15
projectonline	15
Dynamicscrmonline	15
Wordonlinebusiness	15
Notifications	12
Onedriveforbusiness	12
Onedrive	12
Msnweather	12
SQL	9
Projectonline (project roadmap)	
Commondataservice	6
Uiflow	6

7.1.3 Labelling

We had the event name labels for each step of the event log that described the activity of that step, but we did not have the connector labelling for each step. There was only knowledge of the connectors for the entire event log, but no details about the connectors for each step were given. Therefore, the labelling for the connectors for each step in the event log was done manually based on the other information such as Step Description and Application Process Name, Automation Step and Label Event Name.

7.1.4 Preprocessing and Splitting

For discovery, we need three main requirements of process mining which are Case ID, Activity and Timestamp. For the case ID, we need a case identifier to distinguish different executions of the same process. So we selected the recording ID as the case ID. Activity should be the names for different steps of the process and make sense during the processing. As mentioned in the previous section, we have a Step Name which can be used as an activity, but this is too general, and the discovered processes need to be more specific for each task and connector. To achieve this, we concatenated the Application Process Name column with the Step Name and set it as an activity. Fig 7.3 is an example of an event log Petri net processed using inductive miner, and the activity names are shown.

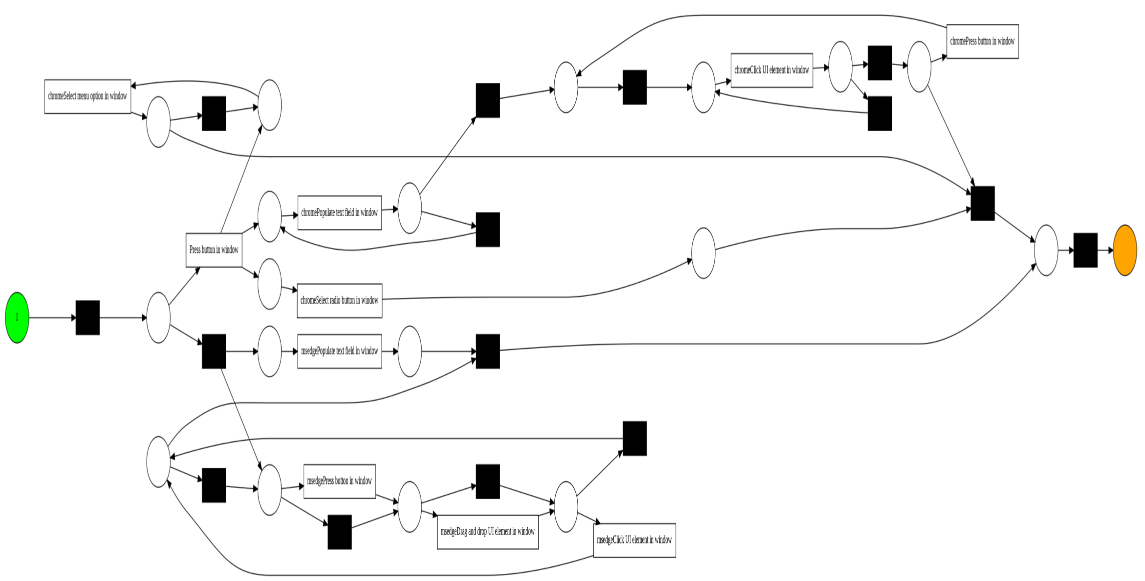

chromeSelect menu option in window
chromePress button in window
chromeClick UI element in window
chromePopulate text field in window
Press button in window
chromeSelect radio button in window
1
msedgePopulate text field in window
msedgePress button in window
msedgeDrag and drop UI element in window
msedgeClick UI element in window

Figure 7.3: An example of visualized petri net of an event log

After labelling the steps, we needed to order the event logs based on the connectors. In order to split the data into training and test folders, first, we needed to group the cases of the event logs based on the connector labels. Therefore, for each connector, a CSV file was created, and all the cases of the event logs with the same connector label were copied to that file.

For splitting, since we had three or six instances of each process, about 20% of these instances with different process IDs were selected for the test, and the rest were chosen for the training procedure. As a result, there were two folders named train and test; the train folder contained files with different connector names with about 80% of the total data of each file, and the test folder contained files with different connector names with about 20% of the total data of each file(Table 7.3).

Table 7.3: Train and Test recording statistics

Connector	Total recording files	Labelled files	Train files	Test files
Googlecalendar	30	26	20	6
Office365users	21	18	14	4
Approvals	30	30	23	7
Microsoftforms	21	21	16	5
Onenote	24	15	11	4
Outlook	21	21	16	5
Planner	21	18	13	5
rss	18	12	8	4
Sendmail	18	18	13	5
Sharepoint	18	15	11	4

7.2 Evaluation and Results

In this section, we will see the results of our solution on the real data to check whether we can answer the questions in section 4. We did the evaluation in two rounds. First round with a smaller size of data, and the second round with about the whole data.

7.2.1 First round of examination

The first round of examination was done on a smaller size of data with just six connectors and selecting one process for train and another one for the test. Each process has three event logs with the connector names labelling. In the first round, we tested all the algorithms for discovery and conformance checking. Tables 7.7, 7.8, 7.9 show the results for inductive miner, heuristic miner and inductive miner infrequent and both alignment and token-based replay conformance checking algorithms. Based on these results, we calculated top-k accuracy for the correct recommendation among the k maximum results.

Table 7.4 contains the top-1 accuracy with the highest value of 83.33% for the heuristic miner as mining algorithm and alignments as the conformance checking. As for the top-2 recommendation, we have the highest accuracy, equal to 100% both for the inductive miner and inductive miner infrequent, along with alignment conformance checking (Table 7.5). Top-3 recommendation accuracy is almost 100% for all the miners using alignment conformance checking (Table 7.6).

We could be able to build a recommendation system that recommends the accurate connector among the top-3 recommendations with an accuracy of 100% based on these results. Therefore, we concluded that we could apply the system to the entire dataset in order to get a comprehensive view of the results.

Table 7.4: Top-1 correct recommendation accuracy for the first round of examinations

Top-1 Accuracy			
Miner / Conformance Checking	**Inductive**	**Heuristic**	**Inductive Infrequent**
Alignment	66.66%	83.33%	66.66%
Token-based Replay	16.66%	66.66%	50%

Table 7.5: Top-2 correct recommendation accuracy for the first round of examinations

Top-2 Accuracy			
Miner / Conformance Checking	**Inductive**	**Heuristic**	**Inductive Infrequent**
Alignment	100%	83.33%	100%
Token-based Replay	50%	66.66%	66.66%

Table 7.6: Top-3 correct recommendation accuracy for the first round of examinations

Top-3 Accuracy			
Miner / Conformance Checking	**Inductive**	**Heuristic**	**Inductive Infrequent**
Alignment	100%	100%	100%
Token-based Replay	83.33%	66.66%	100%

Train models \ Test logs	Projectonline	Powerbi	Planner	Outlook	Microsoftforms	Approvals.
Projectonline	0.6072	0.2723	0.5314	0.2289	0.2058	0.6289
Powerbi	0.4226	0.8296	0.4047	0.3792	0.7222	0.2669
Planner	0.8888	0.4405	0.9259	0.4775	0.5643	0.9424
Outlook	0.0606	0.4764	0.0768	0.6555	0.4578	0.2184
microsoftforms	0.1818	0.6277	0.2601	0.3916	0.8406	0.3566
approvals	0.5893	0.4116	0.6023	0.5395	0.5488	0.7463
Max	0.8888	0.8296	0.9259	0.6555	0.8406	0.9424
Max index (Rank)	2	1	1	1	1	2

(a) Inductive miner and alignment conformance checking

Train models \ Test logs	projectonline	Powerbi	Planner	Outlook	Microsoftforms	Approvals
Projectonline	0.7479	0.7512	0.7471	0.7034	0.4552	0.7691
Powerbi	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Planner	0.9345	0.9528	0.9435	1.0000	0.9020	0.9808
Outlook	0.4444	0.7162	0.4778	0.8907	0.8435	0.6378
microsoftforms	0.3167	0.8948	0.5831	0.5983	0.9260	0.7538
approvals	0.8073	0.7986	0.8536	0.8427	0.8662	0.9126
MAX	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Max Index (Rank)	4	1	2	3	2	3

(b) Inductive miner and token-based replay conformance checking

Table 7.7: 7.7a: Conformance checking results for Inductive Mining and Alignment conformance checking. 7.7b: Conformance checking results for Inductive mining and token-based replay conformance checking. The columns represent the conformance checking fitness values for alignment method and trace fitness for token-based replay method. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Test logs / Train models	projectonline	Powerbi	Planner	Outlook	Microsoftforms	Approvals
Projectonline	0.7205	0.2291	0.5400	0.2611	0.1630	0.6617
Powerbi	0.4438	0.7247	0.3496	0.3500	0.5220	0.2377
Planner	0.6981	0.4500	0.8073	0.4180	0.6368	0.6689
Outlook	0.1454	0.4250	0.1028	0.5820	0.3765	0.1275
microsoftforms	0.2000	0.5000	0.1906	0.5238	0.8510	0.2959
approvals	0.5168	0.3691	0.5255	0.3889	0.3187	0.4439
MAX	0.7205	0.7247	0.8073	0.5820	0.8510	0.6689
Max Index (Rank)	1	1	1	1	1	3

(a) Heuristic Mining and alignment conformance checking

index	projectonline	Powerbi	Planner	Outlook.csv	Microsoftforms	Approvals
Projectonline	0.8611	0.7083	0.6976	0.7000	0.1795	0.5844
Powerbi	0.7778	0.7917	0.7143	0.7222	0.6250	0.4702
Planner	0.7421	0.6500	0.6361	0.7500	0.2222	0.5872
Outlook	0.7500	0.6611	0.7500	0.8194	0.5152	0.3028
microsoftforms	0.1944	0.6143	0.3115	0.5000	0.6778	0.6951
approvals	0.7143	0.6361	0.6917	0.6667	0.3081	0.4539
MAX	0.8611	0.7917	0.7500	0.8194	0.6778	0.6951
Max Index (Rank)	1	1	5	1	1	5

(b) Heuristic Mining and Token-based Replay conformance checking

Table 7.8: 7.8a: Conformance checking results for Heuristic Mining and Alignment conformance checking. 7.8b: Conformance checking results for Heuristic mining and token-based replay conformance checking. The columns represent the conformance checking fitness values for alignment method and trace fitness for token-based replay method. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Train models \ Test logs	projectonline	Powerbi	Planner	Outlook	Microsoftforms	Approvals
Projectonline	0.7205	0.2738	0.6043	0.3222	0.2289	0.7184
Powerbi	0.3907	0.6541	0.3706	0.3835	0.5485	0.2051
Planner	0.9023	0.5053	0.9041	0.4694	0.5622	0.8764
Outlook	0.0333	0.4375	0.0478	0.7010	0.4452	0.1760
microsoftforms	0.1666	0.5791	0.2408	0.4285	0.7384	0.3141
approvals	0.6129	0.3583	0.5977	0.5132	0.5364	0.7415
Max	0.9023	0.6541	0.9041	0.7010	0.7384	0.8764
Max Index (Rank)	2	1	1	1	1	2

(a) Inductive Mining infrequent(default parameters) and alignment conformance checking

Train models \ Test logs	projectonline	Powerbi	Planner	Outlook	Microsoftforms	Approvals
Projectonline	0.9447	0.8710	0.9351	0.8622	0.8353	0.9292
Powerbi	0.9770	0.9020	0.9286	0.8614	0.8353	0.7603
Planner	0.9462	0.8892	0.9685	0.8219	0.8343	0.9633
Outlook	0.5000	0.7770	0.5556	0.9597	0.8784	0.7278
microsoftforms	0.8077	0.9145	0.8517	0.7842	0.9264	0.8668
approvals	0.8363	0.8406	0.8736	0.8690	0.8958	0.9289
Max	0.9770	0.9145	0.9685	0.9597	0.9264	0.9633
Max Index (Rank)	3	2	1	1	1	3

(b) Inductive Mining infrequent (default parameters) and token-based replay conformance checking

Table 7.9: 7.9a: Conformance checking results for Inductive mining infrequent and alignment conformance checking. 7.9b: Conformance checking results for Inductive mining infrequent and token-based replay conformance checking. The columns represent the conformance checking fitness values for alignment method and trace fitness for token-based replay method. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

7.2.2 Second round of examinations

In this section, we examine our recommender system on the entire data mentioned in section 7.1. But for the connectors, we selected the top-10 frequent ones among the connectors in table 7.2. Also, the numbers of the train and test recording files for each connector were indicated in table 7.3. As a result of the poor results from the token-based replay conformance checking algorithm in the first round, we did not use it in the second round. But we changed the noise threshold parameter of inductive mining infrequent, and we got better results. The noise threshold indicates how much behavior can be ignored by the miner. The default value for the noise threshold in PM4PY is 0.

The results that we can see in tables 7.10,7.11 and 7.12 are the results that we obtained from tables 7.13,7.14,7.15,7.16. As we can see from the table, we were able to achieve a 60% accuracy for the top-1 recommendation, a 70% accuracy for the top-2 recommendation, and an 80% accuracy for the top-3 recommendation of the correct connector. According to the top-1 recommendation, inductive miner infrequent had the best performance with a noise threshold of 0.5. For the top-2 and top-3 recommendations, inductive miner and inductive miner infrequent were equal in performance, but overall based on the detailed results in conformance checking values, we can say that inductive miner infrequent is the best algorithm. The problem with heuristic miner can be that it can not guarantee a sound model and when we calculate the conformance checking, this can lead to low accuracy.

Table 7.10: Top-1 correct recommendation accuracy for the second round of examinations

Top-1 Accuracy				
Miner / Conformance Checking	**Inductive**	**Heuristic**	**Inductive Infrequent**	**Inductive Infrequent (nt:0.5)**
Alignment	40%	10%	40%	60%

Table 7.11: Top-2 correct recommendation accuracy for the second round of examinations

Top-2 Accuracy				
Miner / Conformance Checking	**Inductive**	**Heuristic**	**Inductive Infrequent**	**Inductive Infrequent (nt:0.5)**
Alignment	70%	20%	70%	70%

Table 7.12: Top-3 correct recommendation accuracy for the second round of examinations

Top-3 Accuracy				
Miner / **Conformance Checking**	**Inductive**	**Heuristic**	**Inductive Infrequent**	**Inductive Infrequent (nt:0.5)**
Alignment	70%	50%	80%	80%

Table 7.13: Conformance checking results for Inductive Mining infrequent (noise threshold : 0.5) and Alignment conformance checking.The columns represent the conformance checking fitness values. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Test logs / Train models	googlecalendar	Office365users	Approvals	Microsoftforms	onenote	Outlook	Planner	rss	Sendmail	Sharepoint
googlecalendar	0.6208	0.501	0.3917	0.4723	0.5826	0.3755	0.3136	0.673	0.593	0.595
Office365users	0.405	0.6729	0.418	0.610	0.5351	0.339	0.4743	0.748	0.499	0.6357
Approvals	0.241	0.595	0.699	0.574	0.479	0.238	0.572	0.561	0.348	0.647
Microsoftforms	0.340	0.5830	0.634	0.6281	0.589	0.372	0.5093	0.733	0.600	0.787
onenote	0.309	0.4767	0.371	0.3917	0.6125	0.341	0.2563	0.585	0.435	0.4502
Outlook	0.578	0.606	0.4593	0.686	0.629	0.7777	0.271	0.607	0.5556	0.781
Planner	0.288	0.364	0.6844	0.3658	0.504	0.166	0.683	0.170	0.3401	0.8772
rss	0.099	0.2293	0.2267	0.326	0.291	0.0	0.2865	0.1590	0.277	0.548
Sendmail	0.5396	0.6370	0.451	0.665	0.6676	0.391	0.4458	0.805	0.7497	0.746
Sharepoint	0.3098	0.5409	0.4206	0.5520	0.6108	0.3716	0.2695	0.5246	0.3806	0.6849
MAX	0.6208	0.6729	0.6999	0.6867	0.6676	0.7777	0.6836	0.8056	0.7497	0.8772
Max Index (Rank)	1	1	1	3	3	1	1	10	1	5

Table 7.14: Conformance checking results for Inductive Mining infrequent (default parameters) and Alignment conformance checking. The columns represent the conformance checking fitness values. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Test logs / Train models	googlecalendar	Office365users	Approvals	Microsoftforms	onenote	Outlook	Planner	rss	Sendmail	Sharepoint
googlecalendar	0.6414	0.5077	0.3798	0.4700	0.5164	0.2853	0.4523	0.7360	0.6378	0.6695
Office365users	0.4789	0.8979	0.4717	0.9148	0.6634	0.4711	0.5309	0.9516	0.845	0.7958
Approvals	0.4789	0.8727	0.9857	0.8737	0.6991	0.4600	0.8431	0.9397	0.7685	0.9494
Microsoftforms	0.5412	0.7458	0.7660	0.8622	0.7884	0.5730	0.6469	0.8921	0.7675	0.9286
onenote	0.4814	0.6003	0.5234	0.4940	0.7180	0.5155	0.2876	0.7941	0.4714	0.5035
Outlook	0.6968	0.7727	0.4955	0.7837	0.7527	0.8652	0.3400	0.8563	0.7783	0.9008
Planner	0.3232	0.4118	0.8299	0.5452	0.5711	0.2005	0.8383	0.1818	0.5868	0.9541
rss	0.1333	0.2615	0.2137	0.3668	0.2916	0.0	0.386	0.2045	0.3332	0.5730
Sendmail	0.6357	0.7943	0.4807	0.8501	0.6641	0.4373	0.5008	0.9433	0.9361	0.7791
Sharepoint	0.4162	0.6508	0.4830	0.6607	0.6784	0.4275	0.3936	0.6675	0.4560	0.8495
MAX	0.6968	0.8979	0.9857	0.9148	0.7884	0.8652	0.8431	0.9516	0.9361	0.9541
Max Index (Rank)	2	1	1	2	3	1	2	9	1	5

Table 7.15: Conformance checking results for Inductive Mining and Alignment conformance checking. The columns represent the conformance checking fitness values. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Test logs / Train models	googlecalendar	Office365users	Approvals	Microsoftforms	onenote	Outlook	Planner	rss	Sendmail	Sharepoint
googlecalendar	0.7780	0.6343	0.4518	0.5977	0.6232	0.4136	0.4810	0.8184	0.7355	0.7220
Office365users	0.5444	0.9544	0.5066	0.9432	0.7884	0.5111	0.5642	0.9635	0.8699	1.0
Approvals	0.5027	0.9663	1.0	0.9623	0.7884	0.4711	0.9047	0.9635	0.8699	1.0
Microsoftforms	0.5700	0.8384	0.7302	0.9566	0.7884	0.5309	0.7240	0.9880	0.8582	0.9722
onenote	0.5027	0.6658	0.4852	0.5537	0.7627	0.4732	0.4166	0.9047	0.6565	0.4815
Outlook	0.8493	0.8735	0.5098	0.8474	0.7527	0.9271	0.4843	0.9040	0.8653	0.9286
Planner	0.3881	0.5684	0.8396	0.6312	0.6322	0.2589	0.9666	0.2380	0.7052	0.9722
rss	0.2222	0.3604	0.2740	0.4068	0.3907	0.0620	0.4543	0.2159	0.4180	0.5730
Sendmail	0.6243	0.7968	0.4915	0.8581	0.6638	0.4589	0.4834	0.8726	0.9393	0.7442
Sharepoint	0.4075	0.8214	0.4302	0.8375	0.7335	0.4198	0.525	0.9397	0.7957	0.9286
MAX	0.8493	0.9663	1.0	0.9623	0.7884	0.9271	0.9666	0.9880	0.9393	1.0
Max Index (Rank)	2	2	1	2	4	1	1	10	1	6

Table 7.16: Conformance checking results for Heuristic Mining and Alignment conformance checking. The columns represent the conformance checking fitness values. The green colors show the correct and red colors show the wrong classification for the test logs. Max row indicates the maximum value in the corresponding column and the Max index shows the ranking of the correct classification among the values in that column.

Test logs / Train models	googlecalendar	Office365users	Approvals	Microsoftforms	onenote	Outlook	Planner	rss	Sendmail	Sharepoint
googlecalendar	0.3965	0.5670	0.2978	0.4813	0.4105	0.4746	0.2137	0.7052	0.5598	0.3337
Office365users	0.2305	0.3615	0.1630	0.3040	0.3239	0.2569	0.0823	0.2430	0.3656	0.3987
Approvals	0.2305	0.5508	0.4394	0.4216	0.3684	0.3028	0.3767	0.2529	0.4756	0.4654
Microsoftforms	0.2218	0.3708	0.5560	0.4103	0.3397	0.5198	0.3923	0.3224	0.2680	0.5413
onenote	0.2211	0.4519	0.2573	0.2836	0.4679	0.3447	0.1678	0.6354	0.1752	0.3202
Outlook	0.4982	0.5370	0.2670	0.4441	0.4452	0.4632	0.1972	0.2544	0.5728	0.4878
Planner	0.1900	0.2159	0.3767	0.3698	0.3440	0.2005	0.3132	0.0341	0.3986	0.6689
rss	0.1900	0.2297	0.1563	0.3378	0.2093	0.1720	0.1758	0.0682	0.4560	0.4080
Sendmail	0.4416	0.3128	0.1870	0.3857	0.3239	0.2464	0.1028	0.2430	0.5409	0.5814
Sharepoint	0.1871	0.3391	0.2161	0.2743	0.3119	0.2120	0.3593	0.2757	0.2924	0.4920
MAX	0.4982	0.5670	0.5560	0.4813	0.4679	0.5198	0.3923	0.7052	0.5728	0.6689
Max Index (Rank)	3	6	2	4	1	3	4	9	3	4

7.3 Limitations

In this section, we discuss the limitations that we faced during the development of this thesis. The limitations are described in this section, and all of them can be solved and re-implemented in the future.

The data we used had just 165 CSV files without the labelling of the connectors for each step in the event logs. Labelling all the steps manually would have some mistakes and problems and would considerably affect the results. In our solution, we tried to label the step acceptably according to other information in that corresponding step, such as the description or application name, but still, there were some steps that we could not label. However, this problem can be solved whether by labelling automatically or using an unsupervised system where there is no need for labels.

Another limitation of the data was the way the users recorded the tasks. The tasks were limited to some business activities, such as sending an email or scheduling a meeting, and they just used a number of connectors for the automation. This has an impact on the generalism of our system, where it could have different results with the selected algorithms on another set of processes.

There are over 100 algorithms for process discovery that are able to discover the processes. Also, there are several conformance checking techniques. Due to our limitation of tools and time, we could just test a small part of them. We used the PM4PY python library for implementing our system, and this library does not support so many process discovery algorithms that exist. Also, process mining tools such as PROM and DISCO were difficult to use for that amount of data.

7.4 Evaluation of Research Questions

This section answers the research questions outlined in section 4.1 of the Problem Statement. The answers to the questions will be based on the evaluation of our results.

RQ1: Are process mining algorithms such as Inductive Miner and Heuristic Miner able to discover desktop recorded events acceptably?

A1: One of the metrics for analyzing a process model is to compare it with the event logs by using conformance checking. Based on our first (7.7a, 7.8a, 7.9a) and second rounds (7.13,7.14,7.15,7.16) of examinations where we calculated the conformance checking between the processed models and the event logs using inductive miner(also inductive infrequent mining) and heuristic mining , the alignment conformance checking values in diagonal axis of the result tables were more than 0.60 for 88.88% of the first round and 67.5% of the second round. When the fitness value is equal to 1, the trace is perfectly fitting the model. This shows that the discovered models were matched to the event logs with a conformance value of more than 0.5, which indicates that the discovery algorithms such as inductive and heuristic miner were able to discover the recorded events adequately.

RQ2: Is it possible to use conformance checking algorithms such as alignments for recommending the best process connectors for automating the processes?

A2: As presented in the previous section 7.2, we used the conformance checking algorithms such as alignments and token-based replay for the first round and the alignments for the second round of examinations. Using inductive infrequent mining with the noise threshold parameter equal to 0.5 in the second round of examinations, we found that with 60% accuracy, we achieved the highest values of conformance for the correct connector selected for the test event logs. Based on these values, it is possible to select their corresponding connectors to recommend the most suitable connector to automate the processes.

RQ3: Can we use the fitness values from conformance checking to recommend top-k best connectors?

Considering a process model and a sequence of activities observed in a log, if the same sequence of activities (or a very similar one) is allowed by the model, then the fitness of the trace is high. As a consequence of this fact and the results from the second round of evaluation, where 80% accuracy was achieved when choosing the top-3 best connectors using maximum values of fitness, we can conclude that we can use fitness values to recommend the top-k appropriate connectors.

8 Conclusion and Future work

8.1 Conclusion

The purpose of this thesis was to design a recommender system using process mining to automatically recommend the best connector for automating the recorded tasks by the users. This thesis validates a solution that recommends the best identified processed model in order to find out which models are appropriate for the definition of an event log based on the information provided in the event log.

In our development, we used two main methods of process mining. The first technique was process discovery which takes an event log as input and generates a process model, and the second was conformance checking to assess the degree of compatibility between event data and a process model. The data we used were event logs containing various information from monitoring and recording a user's desktop while performing tasks published by Microsoft.

One of the important parts of our research was finding the proper process discovery algorithms to appropriately discover the desktop recorded event logs. There are several algorithms to test, but we selected some well-known ones, such as Inductive Miner, Heuristic Miner and Inductive Miner Infrequent. As we were able to identify Inductive Miner Infrequent as the best-performing algorithm, we also discovered that we could improve it by modifying the parameters, such as the noise threshold, to achieve even better results.

The second key metric of our system was conformance checking. While using it, you can observe how well an event log and a system net fit together. Also, the advantage of conformance checking is that it can be used to evaluate the results of process discovery. We needed numerous analytics for the examinations to determine whether we could correlate the appropriate process models to the test event logs. The fitness metric of conformance checking allowed us to evaluate our models on test event logs. The conformance checking algorithms used in our research were Alignments and Token-based Replay, where the best-performing one was alignments.

The evaluation process took place in two rounds, where in the first round, the smaller size of data was chosen both for train and test event logs and the connectors. Based on the results in section 7.2.1, we had the best accuracy for top-1 recommendation using Heuristic miner and alignment conformance checking. However, since we just selected one process for each of the train and test data, the results were not accurate.

Then for the second round of examinations in section 7.2.2, we achieved the comprehensive results where we had the best accuracy for the inductive miner infrequent and alignment conformance checking by getting 80% of accuracy for the top-3 recommendation of the accurate connectors.

Hence the results delivered more results such as the importance of the data and the behaviour inside the processes, where by increasing the noise threshold in inductive miner infrequent we could achieve better performance. The provided solution was supervised since we used the labelled data and made process models out of them. Hence a user should have the processed model before starting the automation of the processes.

The GitHub repository with the implemented codes developed is available in the following link with the name ”Recommender-System-Based-on-processMining” [1].

8.2 Future Work

It has been discussed in section 7.3 that there are many limitations, hence so many improvements we can make to obtain better results and accuracy.

According to the data collected, it looks like we can have more event logs by creating some desktop records by using desktop process automation tools to produce more event logs. The number of instances of each process increases by having more data, and this can help us improve our models. The automation we implement should also include a greater number of connectors. With more and varying connectors, we should be able to gain a deeper understanding of the results, trying to improve the outcomes.

The process of labelling the steps in the event logs was one of the most time-consuming parts of our project. This can be improved by either automating the labelling procedure during the automation of processes or trying to use machine learning methods such as clustering, which is unsupervised learning.

We were only able to achieve an accuracy of 80% for the top-3 recommendations, as described in section 7.2. These results can also be improved through the application of other procedural and declarative process discovery methods. While this thesis only uses a few well-known process mining methods, a wide range of methods for process discovery and conformance checking could be selected to improve performance and results.

Another way that you can achieve better results is by changing the parameters of the algorithms. For example, by changing the noise threshold for the inductive miner infrequent we observed an improvement in this project. There is a possibility to find the optimal parameter setting in the context of process mining by applying an adapted version of the *k-fold-cv* set-up, which is an adapted version of the *k-fold-cv* in machine learning [44].

[1] https://github.com/nikraftarf/Recommender-System-Based-on-processMining

Bibliography

[1] Wil Van Der Aalst. *Process mining: data science in action*. Vol. 2. Springer, 2016.

[2] Stefan Z Jovanović, Jelena S Đurić, and Tatjana V Šibalija. “Robotic process automation: overview and opportunities”. In: *International Journal Advanced Quality* 46.3-4 (2018), pp. 34–39.

[3] *Process Advisor*. URL: https://powerautomate.microsoft.com/en-us/process-advisor/.

[4] Wil Aalst and H.M.W. Verbeek. “Process Discovery and Conformance Checking Using Passages”. In: *Fundamenta Informaticae* 131 (Jan. 2014). DOI: 10.3233/FI-2014-1006.

[5] Marc Kerremans et al. “Market guide for process mining”. In: *Gartner Inc* (2018).

[6] Lars Reinkemeyer. “Process mining in action”. In: *Process Mining in Action Principles, Use Cases and Outloook* (2020).

[7] Martin Quinn and Erik Strauss. *The routledge companion to accounting information systems*. Routledge, 2018.

[8] Volodymyr Leno et al. “Robotic process mining: vision and challenges”. In: *Business & Information Systems Engineering* 63.3 (2021), pp. 301–314.

[9] *Connectors overview*. URL: https://docs.microsoft.com/en-us/connectors/connectors. (accessed: 06.07.2022).

[10] Mitchell Pearson et al. “Introduction to Power Automate”. In: *Pro Microsoft Power Platform*. Springer, 2020, pp. 73–78.

[11] Charu C Aggarwal et al. *Data mining: the textbook*. Vol. 1. Springer, 2015.

[12] Deuk Hee Park et al. “A literature review and classification of recommender systems research”. In: *Expert systems with applications* 39.11 (2012), pp. 10059–10072.

[13] Prem Melville and Vikas Sindhwani. “Recommender systems.” In: *Encyclopedia of machine learning* 1 (2010), pp. 829–838.

[14] Jan Recker et al. “Business process modeling-a comparative analysis”. In: *Journal of the association for information systems* 10.4 (2009), p. 1.

[15] Stefan Schönig and Stefan Jablonski. “Comparing declarative process modelling languages from the organisational perspective”. In: *International Conference on Business Process Management*. Springer. 2016, pp. 17–29.

[16] Anne Rozinat et al. “The Need for a Process Mining Evaluation Framework in Research and Practice”. In: *Business Process Management Workshops*. Ed. by Arthur ter Hofstede, Boualem Benatallah, and Hye-Young Paik. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 84–89. ISBN: 978-3-540-78238-4.

[17] Sebastian Dunzer et al. “Conformance checking: a state-of-the-art literature review”. In: *Proceedings of the 11th international conference on subject-oriented business process management*. 2019, pp. 1–10.

[18] Andrea Burattin et al. “Online conformance checking using behavioural patterns”. In: *International Conference on Business Process Management*. Springer. 2018, pp. 250–267.

[19] Josep Carmona et al. “Conformance checking”. In: *Switzerland: Springer.[Google Scholar]* (2018).

[20] Wil Van Der Aalst. “Process mining: Overview and opportunities”. In: *ACM Transactions on Management Information Systems (TMIS)* 3.2 (2012), pp. 1–17.

[21] Anne Rozinat and Wil MP Van der Aalst. “Conformance checking of processes based on monitoring real behavior”. In: *Information Systems* 33.1 (2008), pp. 64–95.

[22] Mansoureh Yari Eili, Jalal Rezaeenour, and Mohammadreza Fani Sani. “A systematic literature review on process-aware recommender systems”. In: *arXiv preprint arXiv:2103.16654* (2021).

[23] Jianmin Wang et al. “On recommendation of process mining algorithms”. In: 2012, pp. 311–318. ISBN: 9780769547527. DOI: 10.1109/ICWS.2012.52.

[24] Joel Ribeiro et al. “A recommender system for process discovery”. In: *International Conference on Business Process Management*. Springer. 2014, pp. 67–83.

[25] Alessandro Terragni and Marwan Hassani. “Analyzing customer journey with process mining: From discovery to recommendations”. In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE. 2018, pp. 224–229.

[26] Nicolas Poggi et al. “Business process mining from e-commerce web logs”. In: *Business process management*. Springer, 2013, pp. 65–80.

[27] Yifan Hu, Yehuda Koren, and Chris Volinsky. “Collaborative filtering for implicit feedback datasets”. In: *2008 Eighth IEEE international conference on data mining*. Ieee. 2008, pp. 263–272.

[28] Seppe KLM Broucke et al. “Uncovering the relationship between event log characteristics and process discovery techniques”. In: *International Conference on Business Process Management*. Springer. 2013, pp. 41–53.

[29] Jochen De Weerdt et al. “A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs”. In: *Information Systems* 37.7 (2012), pp. 654–676.

[30] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *The Journal of Machine learning research* 7 (2006), pp. 1–30.

[31] Wil MP Van Der Aalst. “Process discovery from event data: Relating models and logs through abstractions”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.3 (2018), e1244.

[32] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. “Discovering block-structured process models from event logs containing infrequent behaviour”. In: *International conference on business process management*. Springer. 2013, pp. 66–78.

[33] Lulu Ma. “How to evaluate the performance of process discovery algorithms”. PhD thesis. Master thesis, Eindhoven University of Technology, Netherlands, 2012.

[34] Adriano Augusto et al. “Automated discovery of process models from event logs: review and benchmark”. In: *IEEE transactions on knowledge and data engineering* 31.4 (2018), pp. 686–705.

[35] Boudewijn F Van Dongen, AK Alves de Medeiros, and Lijie Wen. “Process mining: Overview and outlook of petri net discovery algorithms”. In: *transactions on petri nets and other models of concurrency II* (2009), pp. 225–242.

[36] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. “Process mining with the heuristics miner-algorithm”. In: *Technische Universiteit Eindhoven, Tech. Rep. WP* 166.July 2017 (2006), pp. 1–34.

[37] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. “Discovering block-structured process models from event logs-a constructive approach”. In: *International conference on applications and theory of Petri nets and concurrency*. Springer. 2013, pp. 311–329.

[38] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. “Replaying history on process models for conformance checking and performance analysis”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.2 (2012), pp. 182–192.

[39] Arya Adriansyah et al. “Alignment based precision checking”. In: *International conference on business process management*. Springer. 2012, pp. 137–149.

[40] Arya Adriansyah, Boudewijn F van Dongen, and Wil MP van der Aalst. “Conformance checking using cost-based fitness analysis”. In: *2011 ieee 15th international enterprise distributed object computing conference*. IEEE. 2011, pp. 55–64.

[41] Alessandro Berti and Wil MP van der Aalst. “Reviving Token-based Replay: Increasing Speed While Improving Diagnostics.” In: *ATAED@ Petri Nets/ACSD*. 2019, pp. 87–103.

[42] Wai Lam Jonathan Lee et al. “Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining”. In: *Information Sciences* 466 (2018), pp. 55–91.

[43] Alessandro Berti, Sebastiaan J Van Zelst, and Wil van der Aalst. “Process mining for python (PM4Py): bridging the gap between process-and data science”. In: *arXiv preprint arXiv:1905.06169* (2019).

[44] AJMM Weijters. “An optimization framework for process discovery algorithms”. In: *Proceedings of the International Conference on Data Mining, Las Vegas, Nevada, USA*. 2011.

A Python Codes

```python
import os
Model_directory = 'Path to your train/test logs'

file_list_model = []   # define file_list to save all csv files
for subdir, dirs, files in os.walk(Model_directory or
    log_directory):
    for file in files:
        if file.endswith('.csv'):
            file_list_model.append((Model_directory or
                log_directory) +file)   # save the filenames in
                file_list

file_list_model
```

Listing A.1: Importing train and test logs

```python
def Function_Name(model,log):
    model = pd.read_csv(model, sep=',',encoding='latin1')
    log = pd.read_csv(log, sep=',',encoding='latin1')
    model_csv= model[['RecordingId','
        ApplicationProcessNameStepName','TimeStamp']]
    log_csv= log[['RecordingId','ApplicationProcessNameStepName'
        ,'TimeStamp']]

    cols = ['case:concept:name','concept:name','time:timestamp']
    model_csv.columns = cols
    model_csv['time:timestamp'] = pd.to_datetime(model_csv['time
        :timestamp'])
    model_csv['concept:name'] = model_csv['concept:name'].astype
        (str)

    log_csv.columns = cols
    log_csv['time:timestamp'] = pd.to_datetime(log_csv['time:
        timestamp'])
    log_csv['concept:name'] = log_csv['concept:name'].astype(str
        )

    model = log_converter.apply(model_csv, variant=log_converter
        .Variants.TO_EVENT_LOG)
    log = log_converter.apply(log_csv, variant=log_converter.
        Variants.TO_EVENT_LOG)

    # process discovery using Inductive Miner
    net, initial_marking, final_marking = inductive_miner.apply(
        model)

    # process discovery using Inductive Miner infrequent
    net, initial_marking, final_marking = im_f.algorithm.apply(
       model,parameters=None)

    # process discovery using Inductive Miner infrequent with
       noiseThreshold equal 0.5
    net, initial_marking, final_marking = im_f.algorithm.apply(
       model,parameters={"noiseThreshold" : 0.5})

    # process discovery using Heuristic miner with default
       parameters
    net, initial_marking, final_marking = heuristics_miner.apply
       (model,parameters={heuristics_miner.Variants.CLASSIC.
       value.Parameters.DEPENDENCY_THRESH: 0.5})


    # Conformance checking using alignments
    aligned_traces = alignments.apply_log(log, net,
       initial_marking, final_marking)

    # Conformance checking using Token-based Replay
    aligned_traces = token_replay.apply(log, net,
       initial_marking, final_marking)

    df = pd.DataFrame(aligned_traces)
    # Fitness value related to alignment
    dff = df["fitness"].mean()

    # trace_fitness value related to token-based replay
    dff = df["trace_fitness"].mean()

    return dff
```

Listing A.2: The main function for the discovery and conformance experiment

```
x = file_list_log
y = file_list_model

data_new=[]
for i in file_list_model:
  row=[]
  for j in file_list_log:
    row.append(Function_Name(i,j))
  data_new.append(row)


df_new = pd.DataFrame(data_new, index=y, columns=x)
df_max = df_new.max()
df_max = pd.DataFrame(df_max)
df_max.columns= ['MAX']
df_max= df_max.transpose()
df = df_new.append(df_max)
```

```
df
```

Listing A.3: Running and getting the results